Instituto de Ciencias Básicas e Ingeniería

# Licenciatura en ciencias computacionales

## "Práctica 1. Análisis de Expresiones Regulares"


**ALUMNO:**

Linares Carrada Jasiel

**PROFESOR:**

EDUARDO CORNEJO VELAZQUEZ

SEMESTRE: 6 GRUPO: 3

# Práctica 1. Análisis de Expresiones Regulares

## ÍNDICE



# 1. IDENTIFICACIÓN

**Nombre de la práctica:** Análisis de Expresiones Regulares
**No. de práctica:** 1
**No. de integrantes máximo por equipo:** 2
**No. de sesiones:** 2
**Integrantes:** Jasiel Linares Carrada

# 2. INTRODUCCIÓN

Una Gramática Formal definida sobre un alfabeto ∑ es una tupla de la forma:
**G = {∑T, ∑N, S, P}**
donde:

- ∑T es un alfabeto de símbolos terminales
- ∑N es un alfabeto de símbolos no terminales
- S es el símbolo inicial de la gramática
- P es un conjunto de producciones gramaticales

Las expresiones regulares son una notación normalizada para representar lenguajes regulares, permitiendo describir con exactitud y sencillez cualquier lenguaje regular.

# 3. OBJETIVO GENERAL

Construir programas en un lenguaje de programación de alto nivel que realicen el análisis de cadenas de símbolos que forman palabras basado en la implementación de estructuras de datos y algoritmos de análisis, además del uso de expresiones regulares.

# 4. OBJETIVOS ESPECÍFICOS

- Identificar el proceso básico de análisis de cadenas de símbolos de un alfabeto
- Implementar estructuras de datos y algoritmos de análisis de cadenas de símbolos
- Implementar expresiones regulares para el análisis de cadenas de símbolos

# 5. EQUIPO Y SOFTWARE

- Lenguaje de programación: Python 3.x
- Software: Visual Studio Code

# 6. DESARROLLO

```
# Práctica 1. Análisis de Expresiones Regulares

# Jasiel Linares Carrada


import re


def analizar_con_estructuras_datos():

    """Análisis usando estructuras de datos y algoritmos"""

    print("=== ANÁLISIS CON ESTRUCTURAS DE DATOS ===")


    simbolos_especiales = {'@', '#', '$', '%', '&', '_'}

    entrada = input("Ingrese las cadenas a clasificar: ")

    lista_cadenas = entrada.split()


    resultados = {

        'Entero': 0,

        'Minusculas': 0,

        'Mayusculas': 0,

        'Identificador': 0,

        'Simbolo': 0,

        'Compuesta': 0
```

```
    }

    for cadena in lista_cadenas:
        tiene_numeros = False
        tiene_letras_min = False
        tiene_letras_may = False
        tiene_simbolos = False
        tiene_guion_bajo = False

        for caracter in cadena:
            if '0' <= caracter <= '9':
                tiene_numeros = True
            elif 'a' <= caracter <= 'z':
                tiene_letras_min = True
            elif 'A' <= caracter <= 'Z':
                tiene_letras_may = True
            elif caracter == '_':
                tiene_guion_bajo = True
            elif caracter in simbolos_especiales:
                tiene_simbolos = True

        # Clasificación
        if tiene_simbolos:
            clasificacion = "Simbolo" if len(cadena) == 1 and cadena in simbolos_especiales else "Compuesta"
        elif tiene_numeros and not tiene_letras_min and not tiene_letras_may and not tiene_guion_bajo:
            clasificacion = "Entero"
        elif tiene_letras_min and not tiene_numeros and not tiene_letras_may and not tiene_simbolos and not tiene_guion_bajo:
```

```
            clasificacion = "Minusculas"
        elif tiene_letras_may and not tiene_numeros and not tiene_letras_min and not tiene_simbolos and not tiene_guion_bajo:
            clasificacion = "Mayusculas"
        elif (tiene_letras_min or tiene_letras_may or tiene_guion_bajo) and not tiene_simbolos:
            clasificacion = "Identificador"
        else:
            clasificacion = "Compuesta"

        resultados[clasificacion] += 1
        print(f"{cadena}: {clasificacion}")

    print("\n--- RESULTADOS ---")
    for categoria, cantidad in resultados.items():
        print(f"{categoria}: {cantidad}")

def analizar_con_expresiones_regulares():
    """Análisis usando expresiones regulares"""
    print("\n=== ANÁLISIS CON EXPRESIONES REGULARES ===")

    # Expresiones regulares definidas
    patrones = {
        'Entero': r'^\d+$',
        'Minusculas': r'^[a-z]+$',
        'Mayusculas': r'^[A-Z]+$',
        'Identificador': r'^[a-zA-Z_][a-zA-Z0-9_]*$',
        'Simbolo': r'^[@#$%&]$',
        'Compuesta': r'.*'
    }
```

```
    entrada = input("Ingrese las cadenas a clasificar: ")
    lista_cadenas = entrada.split()

    resultados = {categoria: 0 for categoria in patrones.keys()}

    for cadena in lista_cadenas:
        clasificada = False
        for categoria, patron in patrones.items():
            if re.match(patron, cadena):
                resultados[categoria] += 1
                print(f"{cadena}: {categoria}")
                clasificada = True
                break

        if not clasificada:
            resultados['Compuesta'] += 1
            print(f"{cadena}: Compuesta")

    print("\n--- RESULTADOS ---")
    for categoria, cantidad in resultados.items():
        print(f"{categoria}: {cantidad}")

def main():
    """Función principal"""
    print("PRÁCTICA 1: ANÁLISIS DE EXPRESIONES REGULARES")

    while True:
```

```
        print("\nSeleccione el método de análisis:")
        print("1. Estructuras de datos")
        print("2. Expresiones regulares")
        print("3. Salir")

        opcion = input("Opción: ")

        if opcion == '1':
            analizar_con_estructuras_datos()
        elif opcion == '2':
            analizar_con_expresiones_regulares()
        elif opcion == '3':
            print("¡Hasta luego!")
            break
        else:
            print("Opción no válida. Intente de nuevo.")

if __name__ == "__main__":
    main()
```

# 7. RESULTADOS

```
PRÁCTICA 1: ANÁLISIS DE EXPRESIONES REGULARES

Seleccione el método de análisis:
1. Estructuras de datos
2. Expresiones regulares
3. Salir
Opción: 1
=== ANÁLISIS CON ESTRUCTURAS DE DATOS ===
Ingrese las cadenas a clasificar: comida  123hola JASIEL jrf@12 18493
comida: Minusculas
123hola: Identificador
JASIEL: Mayusculas
jrf@12: Compuesta
18493: Entero

--- RESULTADOS ---
Entero: 1
Minusculas: 1
Mayusculas: 1
Identificador: 1
Simbolo: 0
Compuesta: 1
```

```
Seleccione el método de análisis:
1. Estructuras de datos
2. Expresiones regulares
3. Salir
Opción: 2

=== ANÁLISIS CON EXPRESIONES REGULARES ===
Ingrese las cadenas a clasificar: @@@ 12894 mipc4 SILLA coma alex123@
@@@: Compuesta
12894: Entero
mipc4: Identificador
SILLA: Mayusculas
coma: Minusculas
alex123@: Compuesta

--- RESULTADOS ---
Entero: 1
Minusculas: 1
Mayusculas: 1
Identificador: 1
Simbolo: 0
Compuesta: 2
```

# 8. CUESTIONARIO

¿Cuál es la utilidad de las expresiones regulares en la identificación de palabras de un lenguaje?

Las expresiones regulares permiten definir patrones de manera concisa y eficiente para identificar cadenas que pertenecen a un lenguaje específico. Facilitan la validación, búsqueda y extracción de patrones en texto, siendo especialmente útiles en análisis léxico de compiladores y procesamiento de lenguajes naturales.

¿Cuáles elementos del lenguaje de programación utilizaste para la implementación de las estructuras de datos? ¿Cómo las utilizaste?

Utilicé:

- **Conjuntos** para almacenar símbolos especiales
- **Diccionarios** para contar resultados y almacenar patrones
- **Listas** para procesar múltiples cadenas
- **Bucles for** para iterar sobre caracteres y cadenas
- **Condicionales if-elif-else** para la lógica de clasificación

¿Cuál es el proceso que seguiste para analizar las cadenas de símbolos?

1. Dividir la entrada en tokens individuales
2. Para cada token, analizar carácter por carácter
3. Identificar tipos de caracteres presentes (dígitos, letras, símbolos)
4. Aplicar reglas de clasificación basadas en la combinación de tipos
5. Contabilizar resultados por categoría

¿Qué elementos del lenguaje de programación te permitieron implementar expresiones regulares? ¿Qué condiciones o recomendaciones se deben seguir al escribir las expresiones regulares?

Utilicé el módulo re de Python. Recomendaciones:

- Usar anchors (^ y $) para coincidencias exactas
- Ser específico con los caracteres permitidos en cada categoría
- Considerar casos edge y validar exhaustivamente
- Documentar las expresiones regulares para mantener legibilidad

# 9. CONCLUSIÓN

La práctica demostró la efectividad de ambos métodos para el análisis léxico. Las estructuras de datos ofrecen control detallado del proceso, mientras que las expresiones regulares proporcionan una solución más concisa y mantenible. La combinación de ambos enfoques permite un análisis robusto de lenguajes regulares.

# 10. BIBLIOGRAFÍA


- Giró, J., Vázquez, J., Meloni, B., Constable, L. (2015). Lenguajes formales y teoría de automátas. Editorial Alfaomega. Argentina.
- Ruiz Catalán, J. (2010). Compiladores: teoría e implementación. Editorial Alfaomega. México.
- Brookshear, J. G. (1995). Teoría de la Computación. Lenguajes formales, autómatas y complejidad. Editorial Addison-Wesley Iberoamericana.